Computing Basics: A Cross-Platform Guide

By: Oameed Noakoasteen

Version 3, 2026.04

from Internet Archive

Content

- **Linux**
  - **Installation and Package Management**
  - **Terminal and Text Editors**
  - **Environment Management**
  - **Remote Connection**
  - **Shell Commands and Scripting**
    - Most Used Commands of bash
      - Directory and File Operations
        - Traversing Directories
        - Creation, Deletion, Copy/Cut and Paste
        - Archiving
        - File Permissions
        - Symbolic Links
      - Environment Variable Operations
      - Process Management
    - Shell Scripting
- **Windows**
  - **Terminal and Text Editor**
- **Appendices**
  - **Version Control**
  - **Remote HPC Clusters**

Linux
Installation and Package Management

Linux Distribution: `Ubuntu`

1. Download the most recent Long-Term Support (LTS) version of Ubuntu.

   ubuntu.com

2. Use `Rufus` on Windows, `Etcher` on Mac (and Windows) or `Startup Disk Creator utility` on Ubuntu to create a bootable flash drive from the downloaded `.iso` file.

   rufus.ie etcher.balena.io

3. Use the bootable flash drive to install Ubuntu.

Package Management: `apt` and `dpkg`

The `apt` Package Manager

```
apt --help

apt update
apt upgrade

apt install <pkg_1> ...  <pkg_n>
apt remove  <pkg_1> ...  <pkg_n>
```

The `dpkg` Package Manager

```
dpkg --help

dpkg -i <pkg> # install debian package
```

To run any command as an administrator, use it in conjunction with the `sudo` command; For example: `sudo apt update`

Oameed Noakoasteen

Linux
Terminal and Text Editors

`Terminal`

Keyboard Shortcuts

```
Ctrl + Alt   + t # open a new terminal window
                 # when no other terminal windows are open

Ctrl + Shift + n # open a new terminal window
                 # when at least one other terminal window is open

Ctrl + Shift + t # open a new tab in the current terminal window
Ctrl + PgUp      # switch to next     tab
Ctrl + PgDn      # switch to previous tab

Ctrl + r         # search for previous commands from history
                 # press 'Esc' to choose the displayed item

Tab              # auto-complete
```

Configurations

```
Initial terminal size -> 95 columns, 25 rows
# found at Preferences > Profiles > unnamed > Text > text appearance
```

Commands

```
echo           <string>   # print <string>

<command>   >  <file>     # redirect the output of <command> to a  new      file named <file>
<command>   >> <file>     # redirect the output of <command> to an existing file named <file>

<command_1> | <command_2> # a pipe is a form of redirection
                          # pipe sends the output of one command to another for further processing

clear                     # clear terminal window

exit                      # close terminal window/tab
```

Oaneed Noakoasteen

Linux
Terminal and Text Editors

GNOME Text Editor

Keyboard Shortcuts

```
Ctrl + Alt   + <letter> # open a new text editor window
                        # when no other text editor windows are open
                        # deifne "custom shortcut"
                        # found at settings > keyboard > keyboard shortcuts
                        # associate shortcut with command "gnome-text-editor"

Ctrl + n                # open a new text editor window
                        # when at least one other text editor window is open

Ctrl + o                # open  a file
Ctrl + w                # close   file

Ctrl + t                # create a new document (new tab)
Ctrl + PgUp             # switch to next     tab
Ctrl + PgDn             # switch to previous tab

Ctrl + s                # save
Ctrl + Shift + s        # save as
```

Configurations

```
show line numbers -> tick
check spelling    -> remove tick
# found at view options

restore session   -> turn off
# found at main menu > preferences > behavior
```

vim

```
vim --help

vim <file>              # open a file named <file>
                        # if <file> doesn't exist, vim will open a new file named <file>
                        # when <file> is opened, user is in 'normal mode'

    TO EDIT
        i               # enter 'insert mode' for editing
        Esc             # exit  'insert mode'

    TO SAVE/QUIT        (NORMAL MODE)
        :w              # save changes
        :q              # quit
        :x              # save changes and quit
        :q!             # quit without saving changes

    TO SEARCH           (NORMAL MODE)
        / <string>
                n       # next
                Shift + n # previous
                Esc     # exit search

    TO COPY/CUT AND PASTE (NORMAL MODE)
        v               # enter 'visual mode'
                y       # copy
                d       # cut
                p       # paser after cursor
```

Oameed Noakoasteen

Linux
Environment Management

`Miniconda`

Installation

Installing Miniconda

conda --help

```
conda search     <pkg>                              # search for package <pkg>

conda env        list                               # list existing environments

conda env        create -f <file>                   # create an environment from a file named <file>
conda            create -n <name> <pkg_1> ... <pkg_n> # create an environment named <name> containing packages <pkg_1> ... <pkg_n>
conda env        export -n <name> > <file>          # export an environment named <name> into a file named <file>
conda            remove -n <name> --all             # remove an environment named <name>

conda clean      -a

conda activate   <name>                             # activate an environment named <name>
                                                    # in a shell script, use 'source activate <name>'

conda list                                          # list packages installed in the active environment
conda install    <pkg_1> ... <pkg_n>                # install packages <pkg_1> ... <pkg_n> in the active environment
conda deactivate                                    # deactivate the active environment
```

Oameed Noakoasteen

Linux
Remote Connection

Secure Shell (SSH)

```
ssh    <username>@<remote address>                          # connect to remote machine

scp -r <source directory/file> <destination directory/file> # copy directories/files between local and remote machines
                                                            # the '-r' flag (recursive) is only required for directories
                                                            # path on the remote machine side: <username>@<remote address>:<path>
```

Authentication: SSH Key

1. `ssh-keygen` # generate an SSH key on the local machine.
   # the SSH public key is saved in the following file: ~/.ssh/id_rsa.pub

2. `ssh-copy-id <username>@<remote address>` # install the SSH public key on the remote machine.
   # the SSH public key is appended to the following file on the remote machine: ~/.ssh/authorized_keys

The Configuration File

1. Open the following config file:

   `~/.ssh/config`

2. Append the following:

```
Host <name>                                 # <name> referes to a name given to this connection by the user
        hostname <remote address>
        user <username>
        port 22
        LocalForward <port_1> localhost:<port_2> # OPTIONAL
                                                 # port_1 refera to a port number on the local  machine
                                                 # port_2 refers to a port number on the remote machine
```

Oameed Noakoasteen

Linux
Shell Commands and Scripting

Most Used Commands of bash

Directory and File Operations

Traversing Directories

```
pwd                 # print working directory
                    # in path structure, the symbol '~'  refers to the user's home directory
                    # in path structure, the symbol '.'  refers to the current directory
                    # in path structure, the symbol '..' refers to the directory one level up from the current directory

ls -lah <directory> # list content of <directory>
                    # without the argument <directory>, this command lists the content of the working directory
                    # the '-l' flag lists content of the working directory in Long Listing Format
                    # the '-a' flag will include hidden directories and files
                    # the '-h' flag gives file sizes in human readable format
                    # single letter flags can be lumped together, e.g., '-lah'

du -sh  <directory> # get the size of a directory
                    # without the argument <directory>, this command gives the size the working directory
                    # the '-s' flag displays only a total for each argument
                    # the '-h' flag prints sizes in human readable format

cd      <directory> # change working directory to <directory>
                    # without the argument <directory>, this command changes the working directory to the user's home directory
```

Oameed Noakoasteen

Linux
Shell Commands and Scripting

Most Used Commands of bash

Directory and File Operations (Cont'd)

Creation, Deletion, Copy/Cut and Paste of Directories and Files

```
mkdir     <directory>                                          # create a  directory named <directory>

>         <file>                                               # create an file named <file>

rm    -rf <directory/file>                                     # delete directory/file
                                                               # the '-r' flag (recursive) is only required for directories
                                                               # the '-f' flag ignores nonexistent files and arguments, never prompt

cp    -r  <source directory/file> <destination directory/file> # copy directory/file from source and paste it to destination
                                                               # the '-r' flag (recursive) is only required for directories

mv        <source directory/file> <destination directory/file> # move, i.e., cut and paste, directory/file from source to destination
```

Archiving

```
tar --help

tar -cvzf <archive> <directory/file> # create an archive named <archive> in the working directory from <directory/file>
                                     # the '-c' flag  creates a new archive
                                     # the '-v' flag verbosely lists files processed
                                     # the '-z' flag filters the archive through gzip
                                     # the '-f' flag usets archive file name

tar -xvzf <archive>                  # extract the archive named <archive> in the working directory
```

Qameed Noakoasteen

Linux
Shell Commands and Scripting

Most Used Commands of bash

Directory and File Operations (Cont'd)

- File Permissions

```
chmod --help
```

- Symbolic Links

```
ln --help

ln -s <target> /usr/local/bin/<name> # <name> will point to <target>
                                     # the '-s' flag creates a symbolic link
```

Environment Variable Operations

```
printenv <variable>          # print the value of an environment variable named <variable>
                             # without the argument <variable>, lists all the environment variables and their values
                             # e.g., PATH, HOME, PWD, SHELL, USER

export   <variable>=<value> # create an environment variable named <variable> with value <value>
```

Oameed Noakoasteen

Linux
Shell Commands and Scripting

Most Used Commands of bash

Process Management

```
top --help

top                                 # launches the top command interface

        Shift + n                   # sort by PID
        Shift + p                   # sort by CPU usage
        Shift + m                   # sort by memory
        Shift + t                   # sort by running time

        Shift + r                   # reverse order of sort

        k         <PID>             # kill process <PID>

        q                           # quit top interface

    TO DEFINE A FILTER
        o <column title>=<value> # filters processes based on <value>
        =                           # exit filter mode
```

Shell Scripting

A shell script is a text file containing a sequence of commands that are executed by a shell interpreter, such as `bash`. It essentially automates tasks that would otherwise require manual input at the terminal.
An executable shell script is created as follows:

1. Create an empty file with extension `.sh`

   ```
   > <shell_script_name>.sh
   ```

2. In the first line, write the shebang followed by the path to the shell interpreter

   ```
   #!/bin/bash
   ```

3. Write all desired shell commands starting from the second line
4. Make the script executable

   ```
   chmod a+x <shell_script_name>.sh
   ```

The shell script can be executed as follows:

```
./<shell_script_name>.sh
```

Oameed Noakoasteen

Windows Terminal and Text Editor

Windows Terminal

Keyboard Shortcuts

```
Win  + <num>      # open a new terminal window
                  # when no other terminal windows are open
                  # defined by adding the terminal to the taskbar
                  # and placing it at location <num>

Ctrl + Shift + n # open a new terminal window
                  # when at least one other terminal window is open

Ctrl + Shift + t # open a new tab in the current terminal window
Ctrl + PgUp       # switch to next     tab, defined in Settings > Actions
Ctrl + PgDn       # switch to previous tab, defined in Settings > Actions

Ctrl + r          # search for previous commands from history
                  # press 'Esc' to choose the displayed item

Tab               # auto-complete
```

Configurations

```
Default Profile                      -> PowerShell             # Settings > Startup
Launch size                          -> 95 columns, 25 rows    # Settings > Startup

Tab switcher interface Style         -> Traditional navigation # Settings > Interaction
Warn when closing more than one tab -> turn off               # Settings > Interaction

Next     tab                         -> Ctrl + PgUp            # Settings > Actions
previous tab                         -> Ctrl + PgDn            # Settings > Actions
```

PowerShell

PowerShell (Microsoft Store)

Windows Subsystem for Linux (WSL)

WSL Documentation

```
wsl  --help
```

Installation and Management

```
wsl --version                          # check WSL version
wsl --status                           # check WSL status
wsl --list       --verbose             # list installed distributions
wsl --list       --online              # list available distributions

wsl --install    <DistributionName>    # default is Ubuntu
                                       # <DistributionName> is optional

wsl --unregister <DistributionName>    # unregister or uninstall a distribution

wsl --update                           # update WSL
```

Usage

```
wsl  --cd ~/ # run  wsl
exit         # exit wsl
```

Oameed Noakoasteen

Windows
Terminal and Text Editor

`Visual Studio Code (VSCode)`

Installation

Visual Studio Code (Microsoft Store)

Keyboard Shortcuts

```
Win  + <num>     # open a new editor window
                 # defined by adding VSCode to the taskbar
                 # and placing it at location <num>

Ctrl + o         # open an existing file
Ctrl + n         # open a new file (new tab)
Ctrl + PgDn      # switch to next     tab
Ctrl + PgUp      # switch to previous tab

Ctrl + w         # close file

Ctrl + s         # save
Ctrl + Shift + s # save as
```

Configurations

press `Ctrl + Shift + p`, search for `settings` and choose `Open User Settings (JSON)`

```
"breadcrumbs.enabled": false

"editor.autoIndent": "none"

"editor.guides.indentation": false

"editor.minimap.enabled": false

"editor.renderControlCharacters": false

"editor.renderWhitespace": "none"

"files.eol": "\n"

"window.restoreWindows": "none"

"window.newWindowDimensions": "inherit"
```

Oameed Noakoasteen

Appendices
Version Control

`Git`

Initial Setup

1. Create an account in **GitHub** or **GitLab**.
2. Configure `Git` on local machine.

```
git config --global core.editor   'vim'
git config --global core.autocrlf false # the default value is 'false'
```

3. Setup authentication using SSH key.

Push a Project Directory to a New Repository

1. `cd` into the project directory.
2.
```
git init                                    # initialize file tracking
git config user.name  <user name>
git config user.email <user email>
git remote add        origin <remote address> # the value of <remote address> is provided by the Git platform
```

3.
```
git add    -A                               # add all files to staging area
git commit -m <message>                     # record changes from the staging area into the local repository
git push                                    # must use with '--set-upstream origin master' for the first push
```

Useful Commands

```
git config --list --file ~/.gitconfig   # list global configurations

git status                              # summary of the changes in the local repository
git reset                               # reset the staging area
git log                                 # history of commits

git clone                               # clones a repository into a newly created directory
```

Oameed Noakoasteen

Appendices
Remote HPC Clusters

Package Management: Environment Modules

```
module list          # list loaded modules
module avail         # list available modules

module load   <pkg>
module unload <pkg>

module purge         # remove all modules
```

Job Scheduler: Slurm and TORQUE/PBS

Slurm

```
sinfo   -N -l                  # display information about the state of partitions and nodes
                               # the flag '-N', prints information in a node-oriented format
                               # the flag '-l', prints more detailed information

sbatch  <shell_script_name>.sh # submit batch job defined in a shell script named <shell_script_name>.sh
                               # use '#SBATCH' to request resources, e.g., #SBATCH --ntasks=16

squeue                         # check the status of jobs

scancel <job_id>
```

TORQUE/PBS

```
qstat -q                       # list available queues

qsub  <shell_script_name>.pbs  # submit batch job defined in a shell script named <shell_script_name>.pbs
                               # use '#PBS' to request resources, e.g., #PBS -l nodes=1:ppn=16

qstat                          # check the status of jobs

qdel <job_id>
```

Oameed Noakoasteen